User Guide for Discrete Adjoint Solver in TAU

Prepared by:

Nicolas Gauger

With contributions by:

Nicolas Gauger, Zhong-Hua Han, Simone Crippa, Caslav Ilic, Stephan Langer, Joël Brezillon, Richard Dwight

DLR Institute of Aerodynamics and Flow Technology

Deutsches Zentrum für Luft- und Raumfahrt e.V. in der Helmholtz-Gemeinschaft

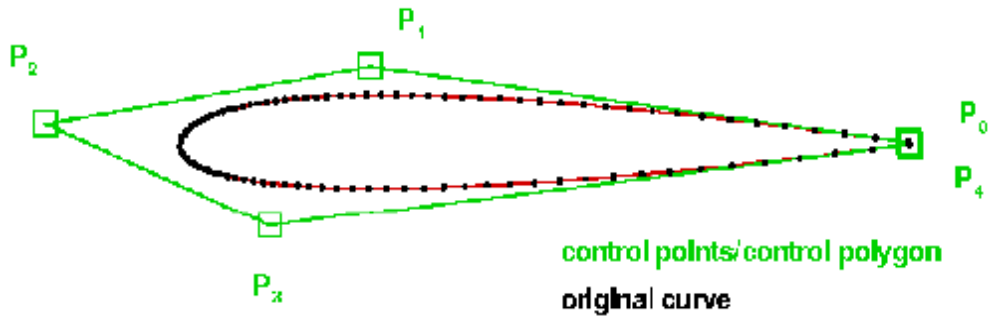
C²A²S²E

Content

- Motivation of adjoint approach by optimization problem
- The adjoint (or dual) equation
- The adjoint approach
- Discrete adjoint solver in TAU (PETSc / FaceMat)
- Installation of TAU for PETSc use
- Input parameters / Settings
- 1. Tutorial / Exercise
- Dual weigted residual (DWR) method for global / local error estimate and functional correction
- 2. Tutorial / Exercise
- 3. Example (Optional)

NOTE: All mentioned files and documents are part of the delivered tar archive

Aerodynamic Shape Optimization

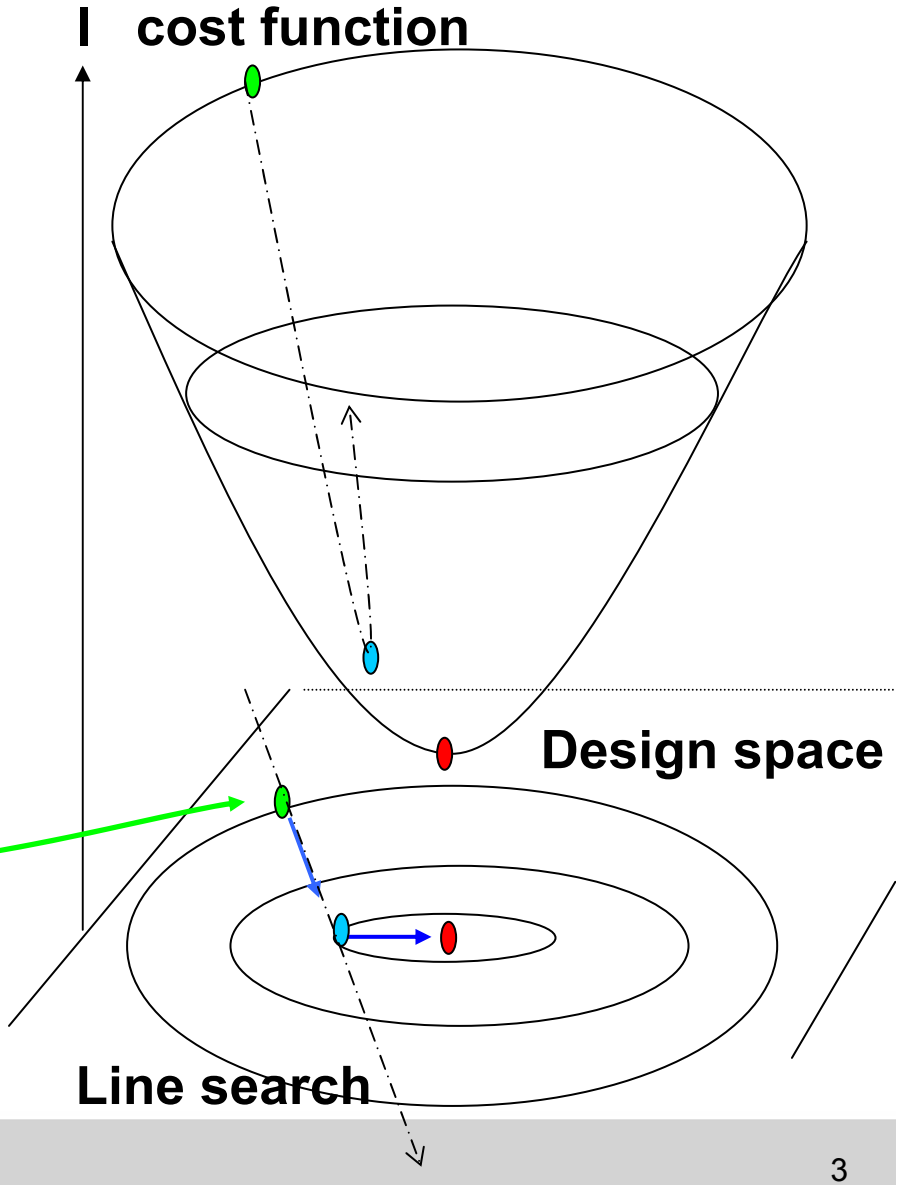


Parameterized airfoil

Vector of (design) parameters *P*

Search direction

“ ↘ ”: $-\nabla I = -\left(\ldots, \frac{\delta I}{\delta P_i}, \ldots \right)^T_{i=1,\ldots,n}$



Finite Differences

i-loop
i=1,...,n

Variation of i-th design variable

$$\delta C_D = \frac{2}{\gamma M_\infty^2 p_\infty C_{ref}} \int_C \delta p \, (n_x \cos\alpha + n_y \sin\alpha) dl$$
$$+ \frac{1}{C_{ref}} \int_C C_p (\delta n_x \cos\alpha + \delta n_y \sin\alpha) dl \, ,$$

Metric sensitivities → pressure variation → aerodynamic sensitivity

i-th component of cost function's gradient

- Finite Differences ⇒ n design variables require n+1 flow calculations

Motivation of Adjoint Approach

High number of design variables

- **Finite Differences** → **n design variables require n+1 flow calculations**

- **Adjoint Approach** → **n design variables require 1 flow and 1 adjoint flow calculation**

Independent of number of design variables

High accuracy

Adjoint Calculus via Lagrangian Function

Optimization problem: $\min_P \; I(U(P)) \quad s.t. \quad R(U(P)) = 0$

Lagrangian function: $L(U,\psi,P) := I(U(P)) + \psi^T R(U(P))$

The Lagrangian multiplier ψ becomes vector of adjoint (or dual) variables for stationary points

Stationary points need to fulfill necessary optimality condition (KKT):

$$\nabla L \overset{!}{=} 0, \quad i.e. \quad \frac{dL}{d(\psi,U,P)} = 0, \quad \Leftrightarrow \quad \frac{dL}{d\psi} = R(U) = 0 \qquad \textbf{(state eq.)}$$

and $$\frac{dL}{dU} = \frac{\partial I}{\partial U} + \psi^T \frac{\partial R}{\partial U} = 0 \qquad \textbf{(dual or adjoint eq.)}$$

and $$\frac{dL}{dP} = \frac{dI}{dP} = \frac{\partial I}{\partial P} + \psi^T \frac{\partial R}{\partial P} = 0 \qquad \textbf{(design eq.)}$$

Adjoint Approach for Sensitivities / Gradients

Solve flow Problem:

$$R(U) = 0$$

Build up adjoint equation and solve for ψ :

$$\frac{\partial I}{\partial U} + \psi^T \frac{\partial R}{\partial U} = 0$$

Use one and the same adjoint solution ψ to calculate the sensitivities / gradient via:

$$\frac{dI}{dP} = \frac{\partial I}{\partial P} + \psi^T \frac{\partial R}{\partial P}$$

Note: We only need to determine the partial derivatives w.r.t. the vector of (design) parameters *P*

There are different ways to calculate the partial derivatives:

$$\frac{dI}{dP} = \frac{\partial I}{\partial P} + \psi^T \frac{\partial R}{\partial P}$$

How to calculate the partials?

There are different ways to calculate the partial derivatives:

$$\frac{dI}{dP} = \frac{\partial I}{\partial P} + \psi^T \frac{\partial R}{\partial P}$$

easy to evaluate, direct calc. by (adj.) solver

here we have two possibilities, see below

First approach: Direct calculation by the (adjoint) solver

→ **available for** α,β,Ma,ρ_∞,T_∞ (see details later on)

Second approach: The „volgrad“ method

→ **solve for** $R(U) = 0,$ **then perturb** *P* **(component by component) and measure the defect in the residual – this defect is the partial derivative** $\frac{dR}{dP}$

How to build up the adjoint equation?

There are two paths to build up and solve the adjoint equation $\frac{\partial I}{\partial U} + \psi^T \frac{\partial R}{\partial U} = 0$:

- **Build up Jacobians and solve adjoint equation by PETSc**
  - **Visit PETSc Homepage:** http://www.mcs.anl.gov/petsc/petsc-as/
  - **Most accurate way to build up Jacobians**
  - **Capable to build up derivatives w.r.t. SA, SAE, Wilcox k-w turbulence models**
  - **Efficient adjoint solve, but memory consuming**
  - **Sequential only**
  - **Recommended for 2D cases**
- **Build up Jacobians and solve adjoint equation by FaceMat option**
  - **Only frozen turbulence and TSL**
  - **Efficient w.r.t. memory, but more expensive w.r.t. run time**
  - **Parallel option**
  - **Recommended for 3D cases**

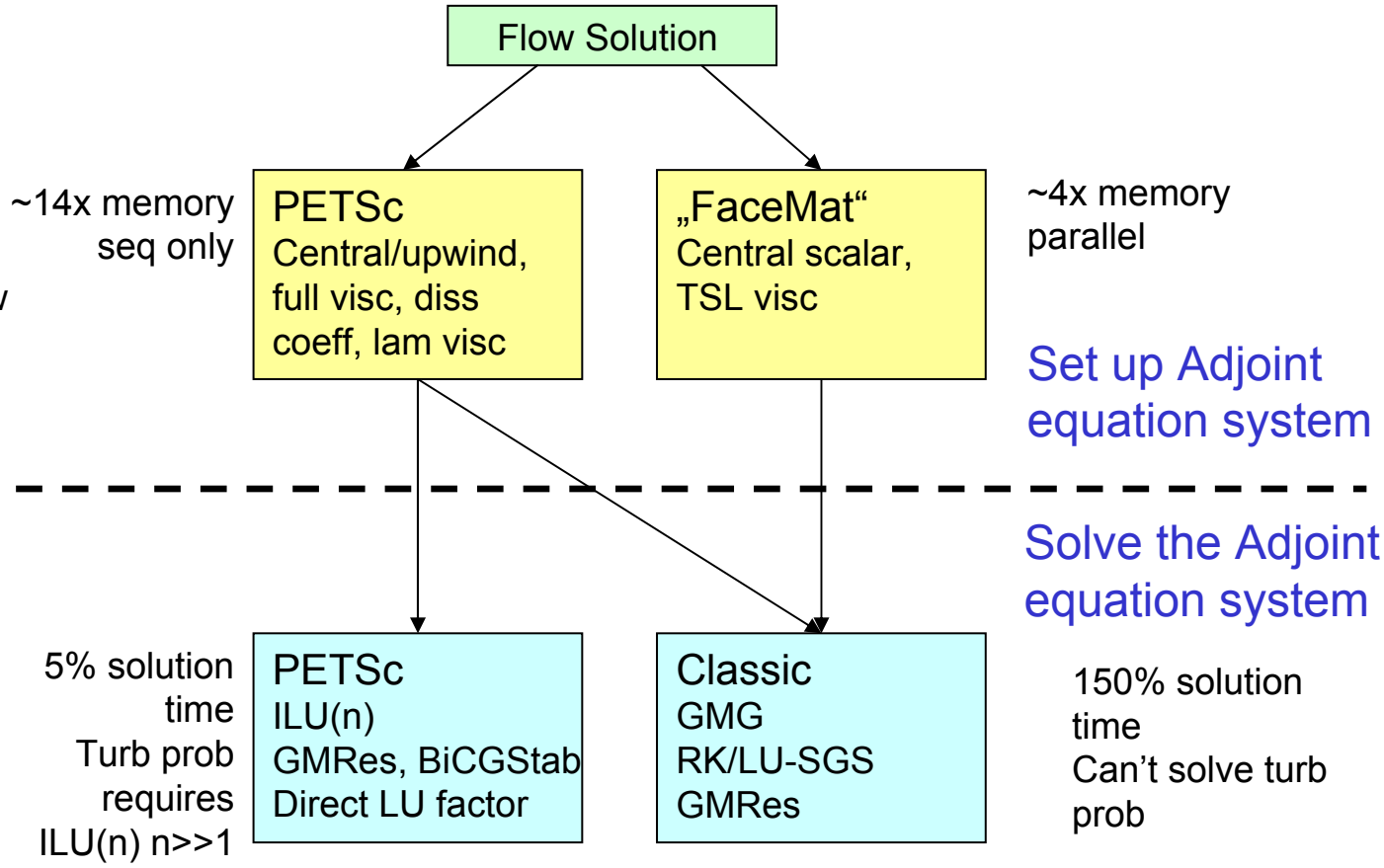
→ See next slide for further details

Discrete Adjoint Solver in TAU

Jacobian

Parameters: from spatial discretization

Both Jacobians:
Turb: 1st-order turb, SA, SAE, Wilcox k-w
BCs: Wall, ffield (no vortex corr), sym



Solution

Parameters: from iterative method, CFL nr etc.

Both Methods:
Parallel

Development, verification & very efficient 2d (Sequential only)

„Production“ Version (3d) (Parallel)

Installation

- Installing TAU will only give you FaceMat, i.e. frozen turb.
- To use PETSc:
  - First install PETSc libraries
    - See guide Compiling_TAU_with_PETSc.pdf to install TAU with PETSc
  - Update your `.taudef` file to include PETSc path
  - Recompile TAU
  - Change „`Linear residual type`“ from default „`Facemat`“ to „`PETSc`“ in input file (details later on)
  - Change „`Jacobian frozen turbulence (0/1)`“ (details later on)
- **Note:** Not possible to install TAU with PETSc and Python!

Input Parameters / Settings for Discrete Adjoint Solver in TAU

Set `Solver type:` **to** `DAdjoint` **for discrete adjoint solver in TAU**

For `Linear residual type:` **choose** `Facemat` **or** `PETSc`

For `PETSc` **choose** `Relaxation solver: PETSc`

NEVER(!) use parallel option or multigrid with `PETSc` **options!**

ONLY(!) use `Central convective turbulence flux: Roe` **with** `PETSc`

Choose your `Cost function:` **as** `C-drag`, `C-lift` **or** `C-my`

(Only) For direct computation of partial derivatives (instead of "volgrad") choose `Design variable:` **(e.g.** `Farfield-alpha` **)**

Input Parameters / Settings for Discrete Adjoint Solver in TAU

With `Facemat` choose only(!) `0` (never(!) `1`) within the following block:

```
Jacobian variables: Cons
Jacobian constant laminar viscosity (0/1): 0
Jacobian frozen turbulence (0/1): 0
Jacobian constant dissipation coeffs (0/1): 0
Jacobian volume scaling (0/1): 0
Jacobian includes timestep (0/1): 0
Jacobian turb. includes timestep (0/1): 0
```

Reason: `Facemat` bases already on certain simplifications (e.g. frozen turbulence)

`0` means off, `1` means on

Now we explain the input parameters of the above given block line by line …

Input Parameters / Settings for Discrete Adjoint Solver in TAU

For `PETSc` options you can choose ...

`Jacobian variables: Cons`
Usually one takes `Cons`, i.e. conservative variables are chosen instead of primitive ones (recommended!)

`Jacobian constant laminar viscosity (0/1):`
When constructing the Jacobian make the simplifying assumption that the laminar viscosity is constant w.r.t. the linearization

`Jacobian frozen turbulence (0/1):`
Stronger than the above assumption; equivalent to the assumtion in `Facemat`

`Jacobian constant dissipation coeffs (0/1):`
Assumption: Coefficients of central dissipation are constant w.r.t. linearization

Input Parameters / Settings for Discrete Adjoint Solver in TAU

For `PETSc` options you can choose ...

`Jacobian volume scaling (0/1):`
Scale the rows of Jacobian by local cell volume (may improve conditioning of the linear system)

`Jacobian includes timestep (0/1):`
Makes a contribution of the reciprocal of the timestep (CFL number based) to the diagonal entries of the Jacobian

`Jacobian turb. includes timestep (0/1):`
As above, but only for turbulence equations

Note: The most accurate way is to choose `PETSc` with `0` for the whole block (recommended!)

Input Parameters / Settings for Discrete Adjoint Solver in TAU

IMPORTANT(!): For use of PETSc you have to add the following line in your parameter file:

PETSc options: -pc_type ilu -pc_factor_levels 4 -ksp_type gmres -ksp_max_it 200 -ksp_rtol 1.e-12 -ksp_monitor

It is highly recommended not to change this options!

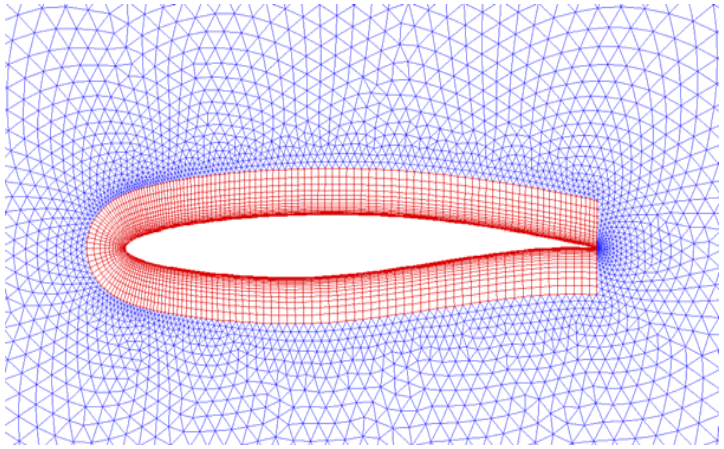
For further details see TAU User Guide user_guide.pdf

and http://www.mcs.anl.gov/petsc/petsc-as/

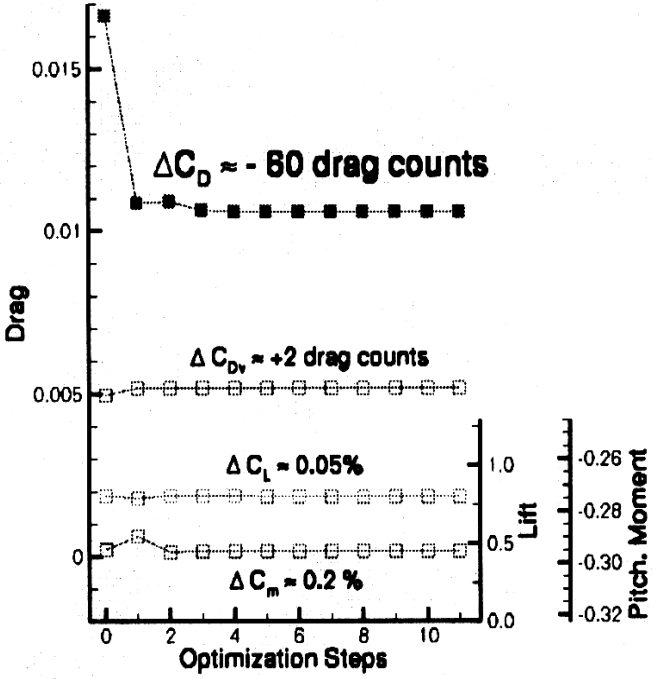
Example: Shape Optimization Based on Discrete Adjoint

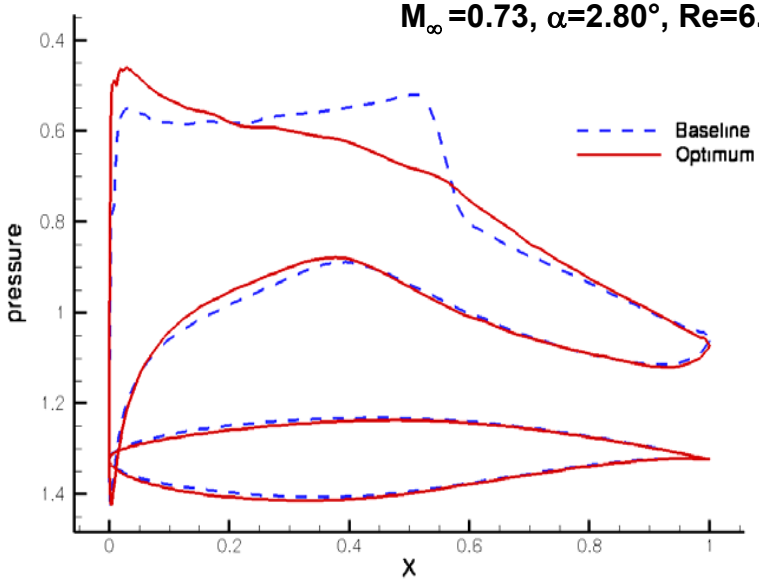
Objective

- **drag minimization for RAE 2822 airfoil at constant lift, pitching moment and AoA**
- **projected steepest descent strategy**
- **flow solver: viscous TAU-Code, SA model**
- **adjoint solver: viscous discret TAU adjoint**



M_∞ =0.73, α=2.80°, Re=6.5x10^6





20 design parameters

1. Example / Tutorial

Adjoint version of Case 9:

The example including grid, parameter files and instructions (README) you will find in the file /1_RAE2822_petsc_kw/

You will learn how to solve $\frac{dC_L}{d\alpha}$ **via (discrete) adjoint simulation, using PETSc and Volgrad**

Just follow the instructions in the README file ...

Later on you can repeat everything with appropriate settings for Facemat option

Have fun and success ;-)

Dual (Adjoint) Weighted Residual Approach (DWR)

h **Fine mesh**

H **Coarse mesh**

U_h^H **State variable** U **extrapolated from coarse to fine mesh**

It holds: $R_h(U_h) = 0$

Then usually we have: $R_h(U_h^H) \neq 0$

In the following, we explain the approach of Dual Weighted Residuals (DWR) for error estimation (in FV context):

- (DWR for FEM: [Rannacher et al., 1997])
- DWR for FV: [Venditti and Darmofal, 2002]:

Dual (Adjoint) Weighted Residual Approach (DWR)

Taylor-series expansion of cost function and residual:

$$I_h(U_h) = I_h(U_h^H) + \left.\frac{\partial I_h}{\partial U_h}\right|_{U_h^H} (U_h - U_h^H) + \ldots \qquad \textbf{(1)}$$

$$R_h(U_h) = R_h(U_h^H) + \left.\frac{\partial R_h}{\partial U_h}\right|_{U_h^H} (U_h - U_h^H) + \ldots \qquad \textbf{(2)}$$

We convert eq. (2) and restrict to 1. order terms:

$$\left(\left.\frac{\partial R_h}{\partial U_h}\right|_{U_h^H}\right)^{-1} (R_h(U_h) - R_h(U_h^H)) \approx (U_h - U_h^H) \qquad \textbf{(3)}$$

$R_h(U_h) = 0$

Substitution of eq. (3) in (1) yields:

$$I_h(U_h) \approx I_h(U_h^H) - \left.\frac{\partial I_h}{\partial U_h}\right|_{U_h^H} \left(\left.\frac{\partial R_h}{\partial U_h}\right|_{U_h^H}\right)^{-1} R_h(U_h^H) \qquad \textbf{(4)}$$

Dual (Adjoint) Weighted Residual Approach (DWR)

It holds:

$$I_h(U_h) \approx I_h(U_h^H) - \left.\frac{\partial I_h}{\partial U_h}\right|_{U_h^H} \left(\left.\frac{\partial R_h}{\partial U_h}\right|_{U_h^H} \right)^{-1} R_h(U_h^H) \quad \textbf{(4)}$$

The adjoint equation reads: $\frac{\partial I}{\partial U} - \psi^T \frac{\partial R}{\partial U} = 0$

Consequently, we have:

$$\left(\psi_h|_{U_h^H} \right)^T = \left.\frac{\partial I_h}{\partial U_h}\right|_{U_h^H} \left(\left.\frac{\partial R_h}{\partial U_h}\right|_{U_h^H} \right)^{-1} \quad \textbf{(5)}$$

Finally, substitution of eq. (5) in (4) yields:

$$I_h(U_h) \approx I_h(U_h^H) - \left(\psi_h|_{U_h^H} \right)^T R_h(U_h^H) \quad \textbf{(6)}$$

Dual (Adjoint) Weighted Residual Approach (DWR)

It holds:

$$I_h(U_h) \approx I_h(U_h^H) - \left(\psi_h \big|_{U_h^H} \right)^T R_h(U_h^H) \qquad \textbf{(6)}$$

We do the further approximation

$$\psi_h \big|_{U_h^H} \approx \psi_h^H$$

and finally we get the error estimate by Venditti and Darmofal

$$I_h(U_h^H) - I_h(U_h) \approx \underline{\left(\psi_h^H\right)^T R_h(U_h^H)}$$

Dual weighted residual (DWR)

Dual (Adjoint) Weighted Residual Approach (DWR)

It turns out, that the repeated extrapolation between certain mesh levels in practice is not handy.

Therefore, instead of $I_h(U_h^H) - I_h(U_h) \approx \left(\psi_h^H\right)^T R_h(U_h^H),$

we consider $I(U_h) - I(U) \approx \psi_h^T R(U_h).$

Now the idea (by R. Dwight) is, to interpret the discretization error $R(U_h)$ **as dissipation error. This yields:**

$$I(U_h) - I(U) \approx \psi_h^T \left(k^{(2)} \frac{\partial R}{\partial k^{(2)}} + k^{(4)} \frac{\partial R}{\partial k^{(4)}} \right)$$

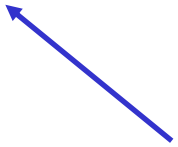
Dual (Adjoint) Weighted Residual Approach (DWR)

The (global) value

$$\psi^T \left(k^{(2)} \frac{\partial R}{\partial k^{(2)}} + k^{(4)} \frac{\partial R}{\partial k^{(4)}} \right),$$

is a scalar product over the computational mesh, i.e. we have a summation over all mesh cells of terms

$$\psi_{i,j}^T \left(k_{i,j}^{(2)} \frac{\partial R_{i,j}}{\partial k^{(2)}} + k_{i,j}^{(4)} \frac{\partial R_{i,j}}{\partial k^{(4)}} \right).$$



Sensor for mesh adaptation
(Local dual weighted residual)

Input Parameters / Settings for Discrete Adjoint Solver in TAU

For adjoint-based error estimation we have the following additional parameters:

Set `Solve dissipation error equation (0/1):` **to** `1`:

This means we assume $R_{i,j}(U_h) \approx k_{i,j}^{(2)} \frac{\partial R_{i,j}}{\partial k^{(2)}} + k_{i,j}^{(4)} \frac{\partial R_{i,j}}{\partial k^{(4)}}$

Note: The original approach of Venditti and Darmofal is not supported in TAU!

Set `Solve linear problem on grid level:` **to appropriate number**

Note: The field output V4 is (dependent on cost function):

$$abs\left[\psi_{i,j}^{T}\left(k_{i,j}^{(2)} \frac{\partial R_{i,j}}{\partial k^{(2)}} + k_{i,j}^{(4)} \frac{\partial R_{i,j}}{\partial k^{(4)}}\right)\right]$$

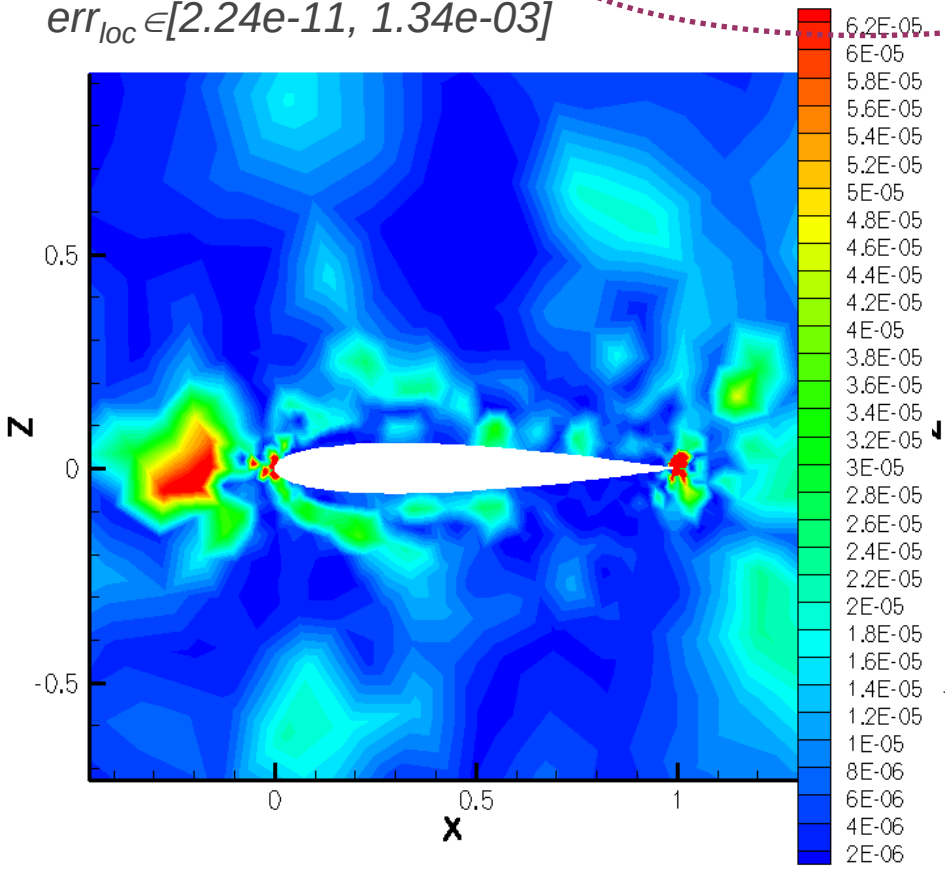
(Local) Adjoint-based Error Estimate / Sensor for Mesh Adaptation by DWR

Field output V4, dependent on cost function

$$abs\left[\psi_{i,j}^{T}\left(k_{i,j}^{(2)}\frac{\partial R_{i,j}}{\partial k^{(2)}}+k_{i,j}^{(4)}\frac{\partial R_{i,j}}{\partial k^{(4)}}\right)\right]$$

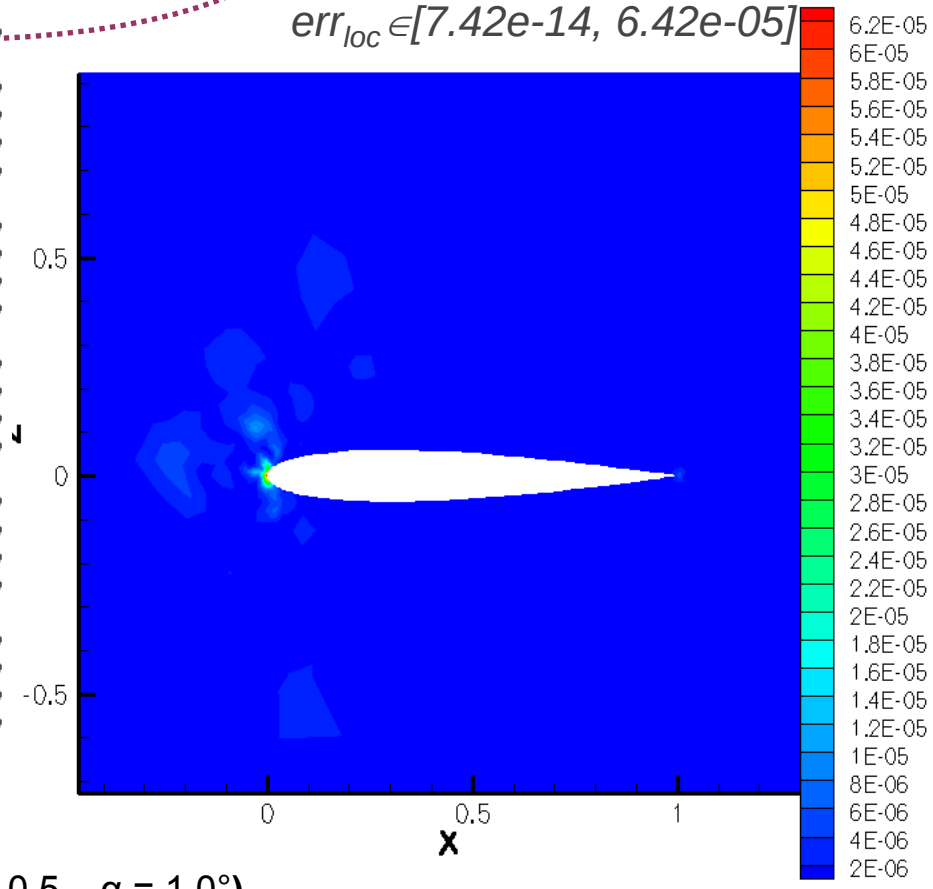
Lift

$err_{loc} \in [2.24e\text{-}11,\ 1.34e\text{-}03]$



Drag

$err_{loc} \in [7.42e\text{-}14,\ 6.42e\text{-}05]$



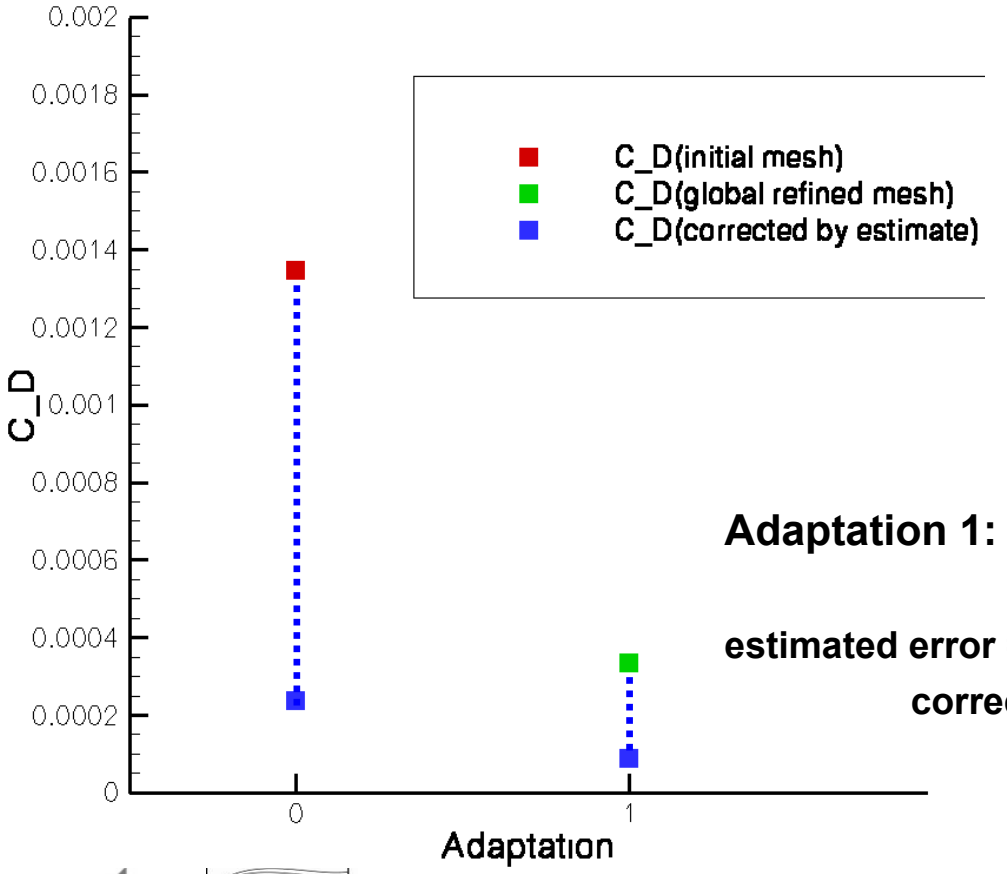
(NACA0012 Ma = 0.5, α = 1.0°)

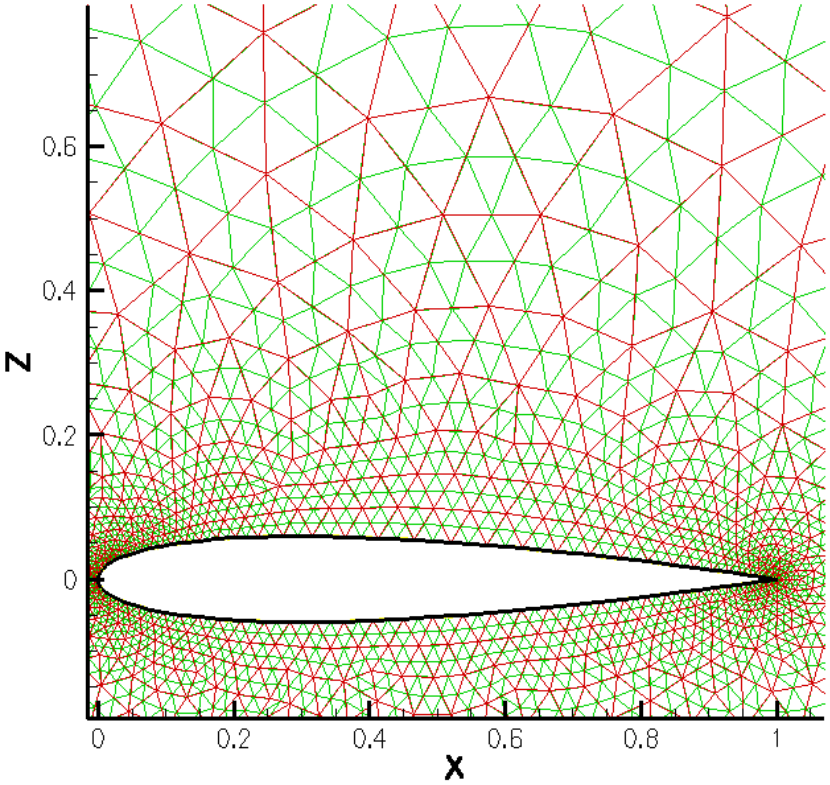
DLR C²A²S²E

Adjoint-based (Global) Error Estimate and Correction

NACA0012 (Euler flow)

Ma = 0.5, α = 1.0°





Adaptation 1:

C_D(global refined mesh) = 0.00033382

estimated error = Ψ*[(dRdk2 * k2) + (dRdk4 * k4)] = - 0.00024721

corrected C_D for global refined mesh = 0.00008661

2. Example / Tutorial

NACA0012 (inviscid):

The example including grid, parameter file and instructions (README) you will find in the file /2_NACA0012_err_estimate/

You will learn how to calculate global error estimates.

Just follow the instructions in the README file ...

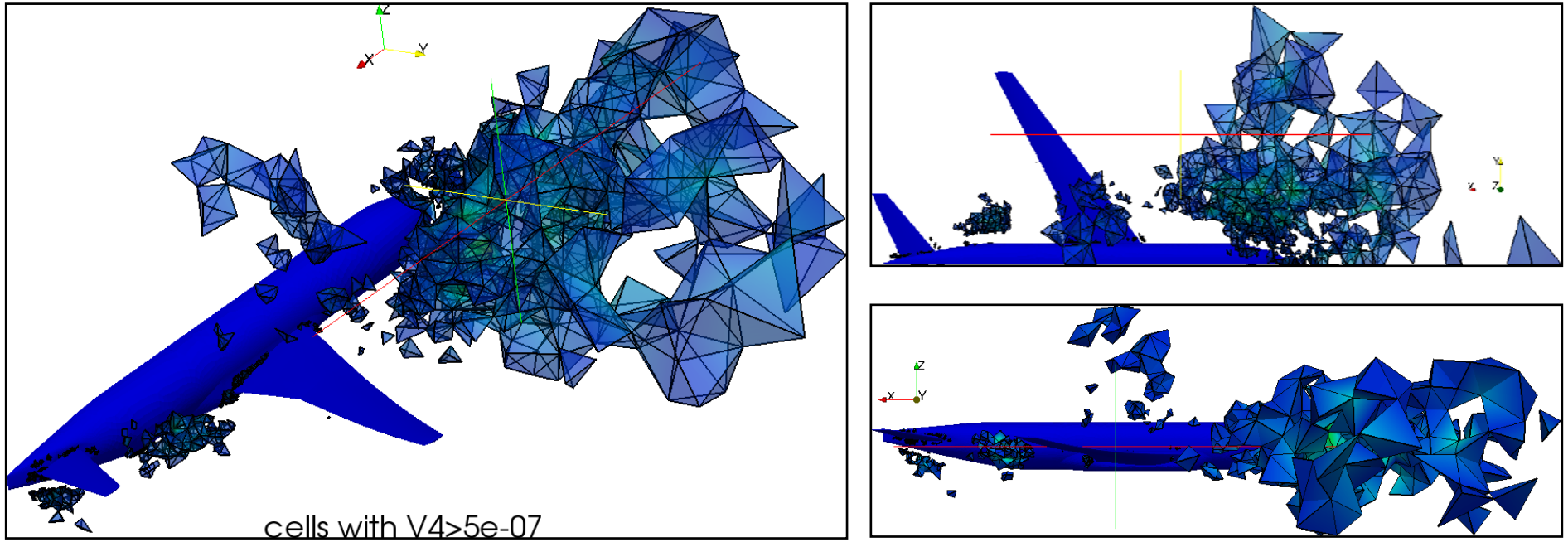
In addition, you learn how to calculate $\frac{dC_D}{d\alpha}$ different from the "volgrag way".

Have fun and success ;-)

(Local) Adjoint-based Error Estimate

- Adjoint-based error estimates coupled to SOLAR
- Manually performed for the DPW4 meshes
- “V4”: Error estimate based on the artificial viscosities k2 and k4
- Only field cells with sensor V4>5e-07 are plotted in color

→ See last/third exercise/file: /3_DPW4_para/



The End